School of Data and Computer Science, Sun Yat-sen University

Maze Problem

16337102 HuangZi lin

September 8, 2018

Contents

1 Task 2

2 Codes 2

3 Results 4

1 Task

- Please solve the maze problem by using BFS or DFS (Python or C++)
- The maze layout can be modeled as an array, and you can use the data file `MazeData.txt` if necessary.
- Please send `E01_YourNumber.pdf` to `ai_2018@foxmail.com`, you can certainly use `E01_Maze.tex` as the LaTeXtemplate.

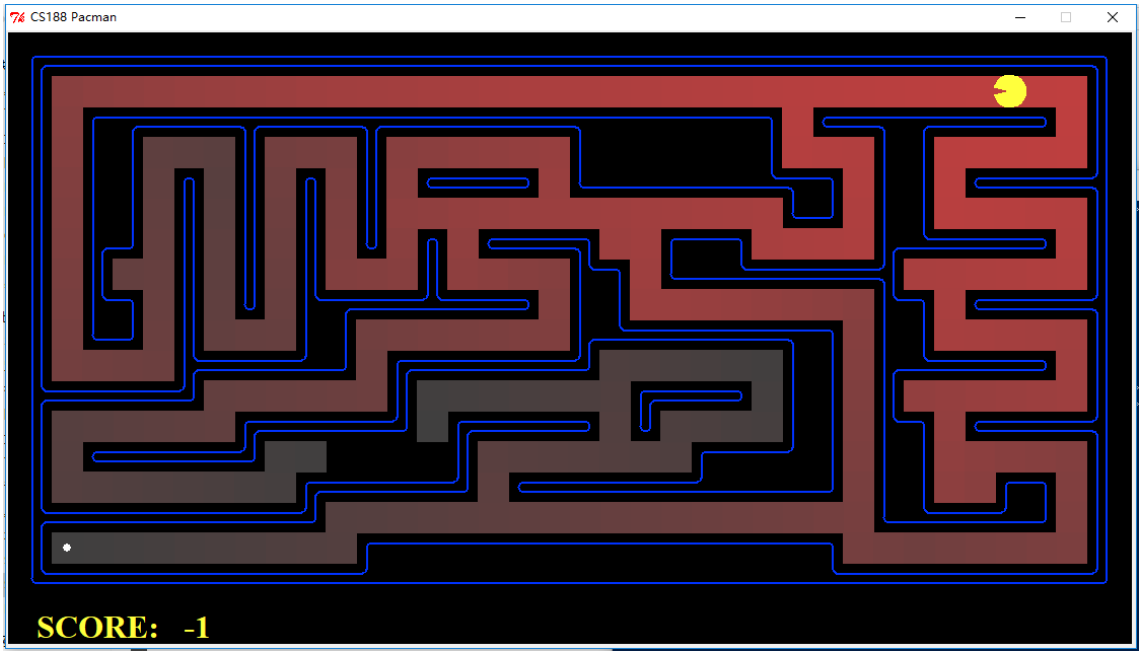

Figure 1: Searching by BFS or DFS

2 Codes

```
import numpy as np

f = open("MazeData.txt", 'r')
data = []
for i in range(18):
    data.append(f.readline())
datas = []
for i in data:
    for n in i:
        if n != '\n':
            datas.append(n)
f.close()

data = [[0 for i in range(36)] for j in range(18)]
for i in range(18):
```

```
    for n in range(36):
        data[i][n] = datas[36 * i + n]

n = 0
start_x = 0
start_y = 0
end_x = 0
end_y = 0

for i in range(18):
    for n in range(36):
        if data[i][n] == 'S':
            start_x = i
            start_y = n
        elif data[i][n] == 'E':
            end_x = i
            end_y = n

path = []
if start_x >= end_x and start_y >= end_y:
    move = np.array([[-1, 0], [0, -1], [1, 0], [0, 1]])
elif start_x >= end_x and start_y <= end_y:
    move = np.array([[-1, 0], [0, 1], [1, 0], [0, -1]])
elif start_x <= end_x and start_y >= end_y:
    move = np.array([[1, 0], [0, -1], [-1, 0], [0, 1]])
else:
    move = np.array([[1, 0], [0, 1], [-1, 0], [0, -1]])

def inPath(x, y, path):
    if [x, y] in path:
        return True
    else:
        return False

def run(start_x, start_y, end_x, end_y, move):
    if start_x == end_x and start_y == end_y:
        return
    elif data[start_x + move[0][0]][start_y+move[0][1]] != '%'
                and not inPath(start_x + move[0][0], start_y + move[0][1], path):
        if start_x == end_x and start_y == end_y:
            return
        start_x += move[0][0]
        start_y += move[0][1]
        path.append([start_x, start_y])
        run(start_x, start_y, end_x, end_y, move)
        start_x -= move[0][0]
        start_y -= move[0][1]
    elif data[start_x + move[1][0]][start_y + move[1][1]] != '%'
                and not inPath(start_x + move[1][0], start_y + move[1][1], path):
```

```
        if start_x == end_x and start_y == end_y:
            return
        start_x += move[1][0]
        start_y += move[1][1]
        path.append([start_x, start_y])
        run(start_x, start_y, end_x, end_y, move)
        start_x -= move[1][0]
        start_y -= move[1][1]
    elif data[start_x + move[2][0]][start_y + move[2][1]] != '%'
                and not inPath(start_x + move[2][0], start_y + move[2][1], path):
        if start_x == end_x and start_y == end_y:
            return
        start_x += move[2][0]
        start_y += move[2][1]
        path.append([start_x, start_y])
        run(start_x, start_y, end_x, end_y, move)
        start_x -= move[2][0]
        start_y -= move[2][1]
    elif data[start_x + move[3][0]][start_y + move[3][1]] != '%'
                and not inPath(start_x + move[3][0], start_y + move[3][1], path):
        if start_x == end_x and start_y == end_y:
            return
        start_x += move[3][0]
        start_y += move[3][1]
        path.append([start_x, start_y])
        run(start_x, start_y, end_x, end_y, move)
        start_x -= move[3][0]
        start_y -= move[3][1]
    else:
        path.pop()
        return

run(start_x, start_y, end_x, end_y, move)
print(path)
print(start_x, start_y)
print(end_x, end_y)
for i in path:
    data[i[0]][i[1]] = '*'

doc = open('out.txt', 'w')
for i in range(18):
    for n in range(36):
        print(data[i][n], end = '', file=doc)
    print('', file=doc)
doc.close()
```

3 Results

```
[[2, 34], [3, 34], [3, 33], [3, 32], [3, 31], [3, 30], [4, 30], [5, 30], [5, 31], [5, 32], [5, 33], [5, 34], [6, 34], [7, 34], [7, 33], [7, 32], [7, 31], [7, 30], [8, 30], [9, 30], [9, 31],
 [9, 32], [9, 33], [9, 34], [10, 34], [11, 34], [11, 33], [11, 32], [11, 31], [11, 30], [12, 30], [13, 30], [14, 30], [14, 31], [13, 31], [13, 32], [13, 33], [13, 34], [14, 34], [15, 34], [
16, 34], [16, 33], [16, 32], [16, 31], [16, 30], [16, 29], [16, 28], [16, 27], [15, 27], [15, 26], [15, 25], [15, 24], [15, 23], [15, 22], [15, 21], [15, 20], [15, 19], [15, 18], [15, 17],
[15, 16], [15, 15], [15, 14], [15, 13], [15, 12], [15, 11], [15, 10], [16, 10], [16, 9], [16, 8], [16, 7], [16, 6], [16, 5], [16, 4], [16, 3], [16, 2], [16, 1]]
```